CIRCUITOS LÓGICOS

Relatório VI

Discente: Rebeca de Macêdo Ferreira. *Matricula:* 2016000524

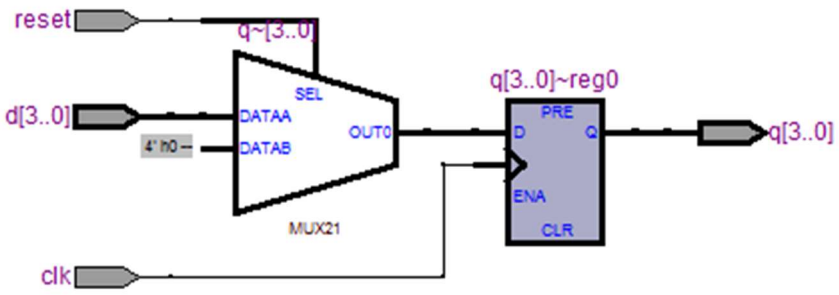
Resettable flip-flop síncrono

Esse projeto diz respeito a um flip-flop com reset sensível ao clock do circuito, o código vhd utilizado também foi do livro Harris & Harris

```
1   library IEEE; use IEEE.STD_LOGIC_1164.all;
2
3   entity floprs is
4       port(clk: in STD_LOGIC;
5            reset: in STD_LOGIC;
6            d: in STD_LOGIC_VECTOR(3 downto 0);
7            q: out STD_LOGIC_VECTOR(3 downto 0));
8   end;
9
10  architecture synchronus of floprs is
11  begin
12      process(clk) begin
13          if clk' event and clk = '1' then
14              if reset = '1' then
15              q <=  "0000";
16              else q <= d;
17              end if;
18          end if;
19      end process;
20  end;
```

Nota-se que aqui, o reset só é testado se o valor lógico do clock for igual a ‘1’

O resultado obtido pelo RTL viewer:



Criamos o código do testbench e fizemos o arquivo test vector

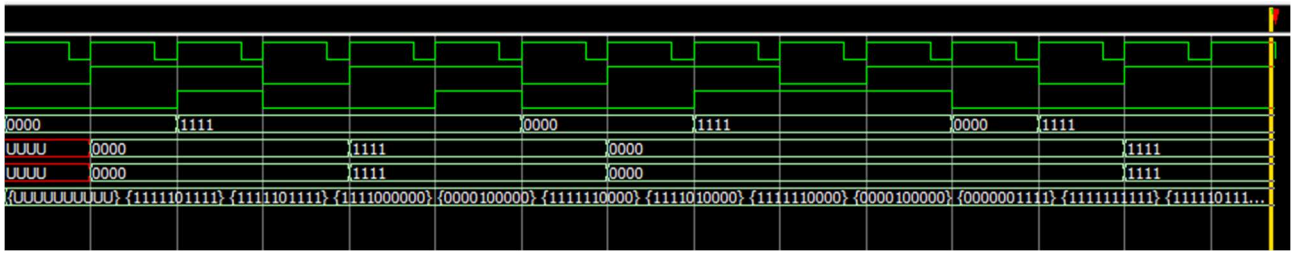
```
6
7  entity testbench_floprs is -- no inputs or outputs
8  end;
9  architecture sim of testbench_floprs is
10     component floprs
11         port(clk: in STD_LOGIC;
12              reset: in STD_LOGIC;
13              d: in STD_LOGIC_VECTOR(3 downto 0);
14              q: out STD_LOGIC_VECTOR(3 downto 0));
15     end component;
16
17 signal clktest: STD_LOGIC;
18 signal clk: STD_LOGIC;
19 signal reset: STD_LOGIC;
20 signal d, q: STD_LOGIC_VECTOR(3 downto 0);
21 signal qexpected: STD_LOGIC_VECTOR(3 downto 0);
22 constant MEMSIZE: integer := 15;
23 type tvarray is array (MEMSIZE downto 0) of
24 STD_LOGIC_VECTOR (9 downto 0);
25 signal testvectors: tvarray;
26 shared variable vectornum, errors: integer;
27 begin
28 -- instantiate device under test
29 dut: floprs port map (clk, reset, d, q);
30 -- generate clock
31 process begin
32     clktest <= '1'; wait for 15 ns;
33     clktest <= '0'; wait for 5 ns;
34 end process;
```

testbench floprs

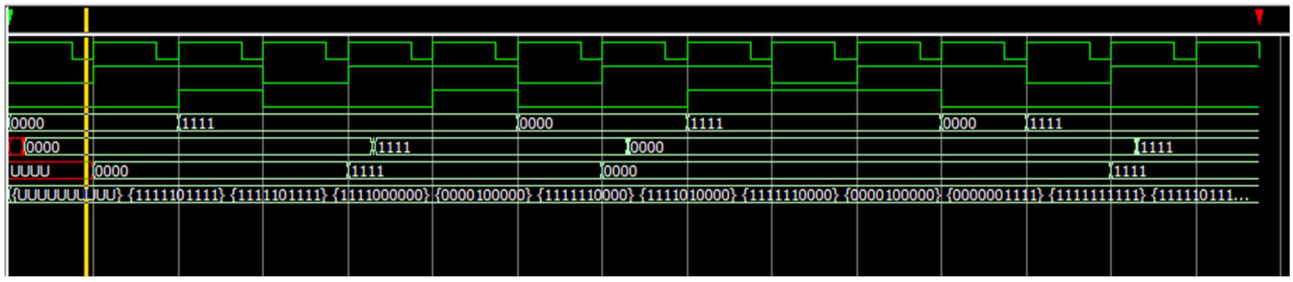
```
1  0000_00_UUUU
2  0000_11_0000
3  1111_11_0000
4  1111_00_0000
5  1111_10_1111
6  1111_11_1111
7  0000_00_1111
8  0000_11_0000
9  1111_11_0000
10 1111_00_0000
11 1111_11_1111
12 0000_11_0000
13 1111_00_0000
14 1111_10_1111
15 1111_10_1111
```

No test vector acima testamos casos em que o clock tem valor alterado pra ‘1’ porém o reset está sendo alterado de ‘0’ para ‘1’ o que faz com que ele não seja reconhecido ainda e portando a saída não é resetada.

Simulamos o RTL level:



Simulamos o Gate level:



Flip-flop com Reset e Enable

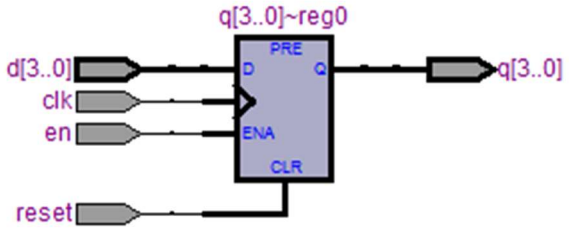
Este projeto é semelhante ao flio-flop anterior, porém agora contamos com um habilitador (enable) além do reset.

Código VHD utilizado:

```
1   library IEEE; use IEEE.STD_LOGIC_1164.all;
2
3   entity flopenr is
4       port (clk,
5             reset,
6             en: in STD_LOGIC;
7             d:  in STD_LOGIC_VECTOR (3 downto 0);
8             q:  out STD_LOGIC_VECTOR (3 downto 0));
9   end;
10
11  architecture asynchronous of flopenr is
12  -- asynchronous reset
13  begin
14      process (clk, reset) begin
15          if reset = '1' then
16              q <= "0000";
17          elsif clk'event and clk = '1' then
18              if en = '1' then
19                  q <= d;
20              end if;
21          end if;
22      end process;
23  end;
```

Pelo código, vemos que a saída será alterada nos sinais de clock ou reset.

Compilamos e o resultado do RTL viewer foi:

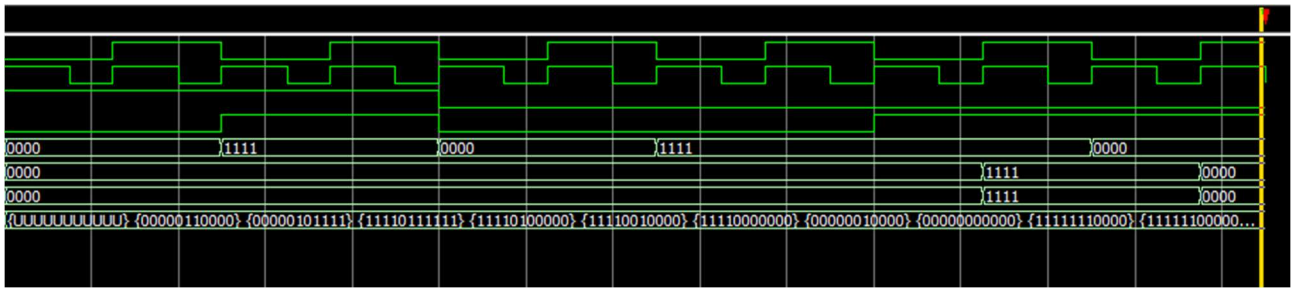


Para as simulações no Modelsim além do testbench, adaptamos um arquivo .tv para vetores de teste (testvector)

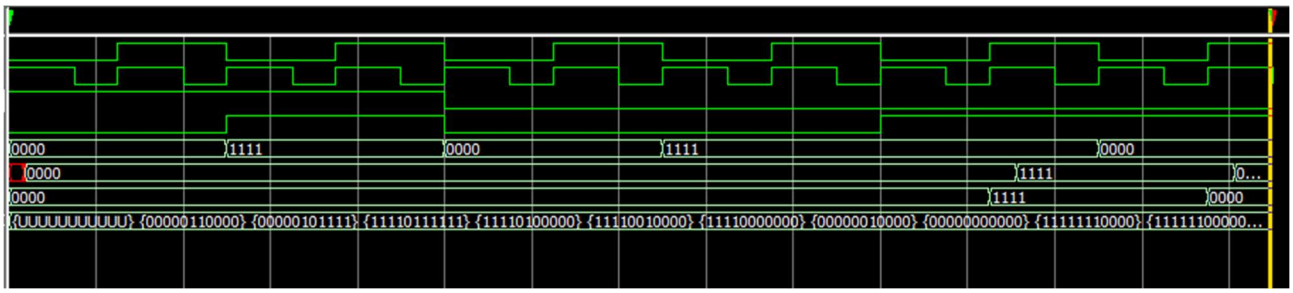
```
1  0000_100_0000
2  0000_101_0000
3  1111_110_0000
4  1111_111_0000
5  0000_000_0000
6  0000_001_0000
7  1111_000_0000
8  1111_001_0000
9  1111_010_0000
10 1111_011_1111
11 0000_010_1111
12 0000_011_0000
```

Com essa tabela verdade e a figura do circuito acima, podemos perceber que a entrada só será copiada pra saída se o clock estiver com valor em ‘1’, o enable habilitado (com valor em ‘1’) e o reset com valor ‘0’; caso o reset esteja com valor ‘1’, a saída será alterada pra ‘0’ ou alterada pra o valor que estiver na entrada caso o enable esteja com valor ‘0’ mas o clock ter mudado pra 1.

Simulação do RTL level:



Simulação do Gate level:



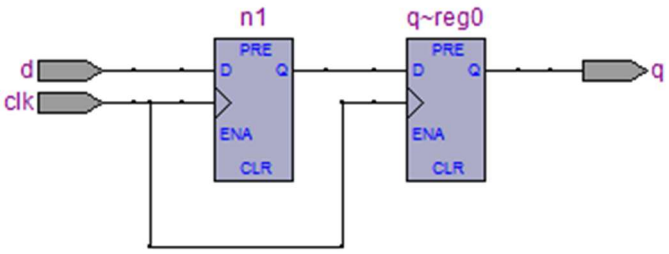
Sincronizador:

O objetivo desse projeto é criar um sincronizador pra sincronizar as entradas assíncronas

Código VHD utilizado:

```
1   library IEEE; use IEEE.STD_LOGIC_1164.all;
2   -- 4.21
3   entity sync is
4       port (clk:  in STD_LOGIC;
5             d:    in STD_LOGIC;
6             q:    out STD_LOGIC);
7   end;
8
9   architecture good of sync is
10      signal n1: STD_LOGIC;
11  begin
12      process (clk) begin
13          if clk'event and clk = '1' then
14              n1 <= d;
15              q <= n1;
16          end if;
17      end process;
18  end;
```

Visualização do RTL viewer:

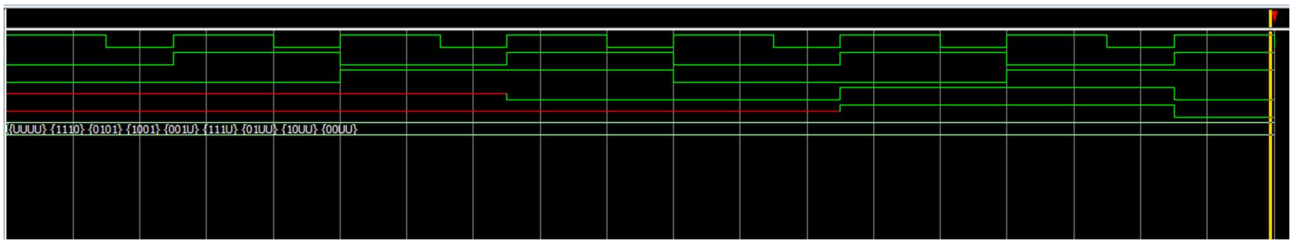


Para simulações com o Modelsim altera construímos um testbench e um tesvector que contém informações de como o circuito deve se comportar

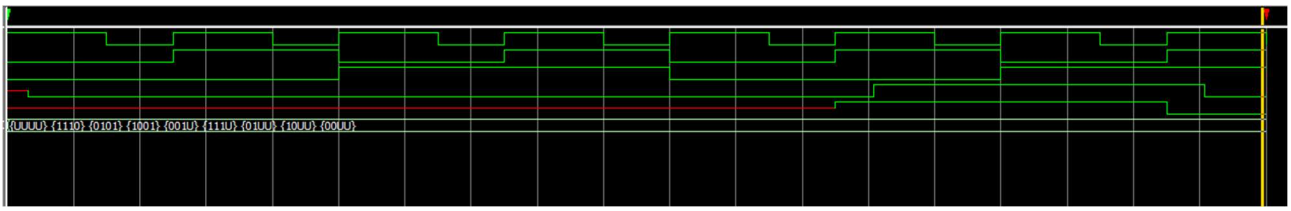
```
1  00UU
2  10UU
3  01UU
4  111U
5  001U
6  1001
7  0101
8  1110
```

Com o testvector acima é possível verificar que nesse circuito que tem dois flip-flops e que a saída do flip-flop n1 será a entrada do próximo flip-flop. Desse modo, quando o clock está em '1' a entrada 'd' é copiada pra saída 'n1' e de 'n1' pra 'q', caso o clock esteja em '0', nada acontece e no começo o valor é indeterminado (U).

Simulação RTL level:



Simulação Gate level:



Latch

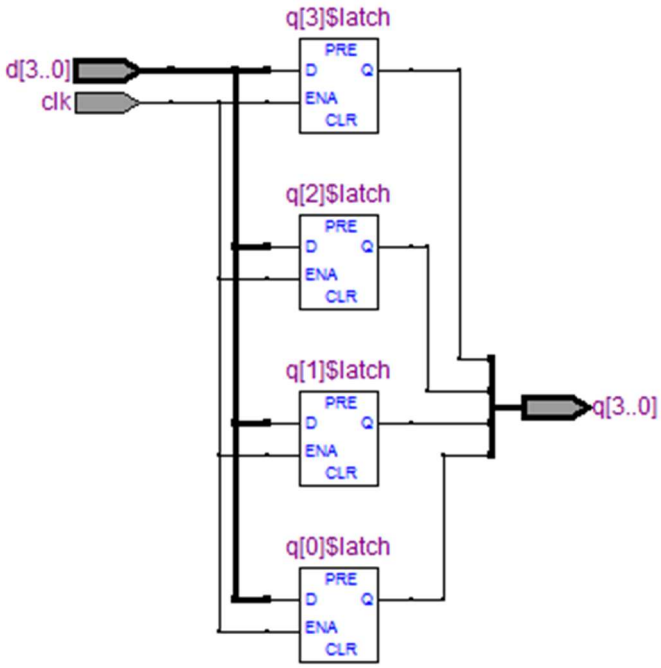
Para este projeto copiamos o código VHD do livro Harris & Harris

```vhdl
library IEEE; use IEEE.STD_LOGIC_1164.all;

entity latch1 is
    port (clk: in STD_LOGIC;
          d: in STD_LOGIC_VECTOR (3 downto 0);
          q: out STD_LOGIC_VECTOR (3 downto 0));
end;

architecture synth of latch1 is
begin
    process (clk, d) begin
        if clk = '1' then
            q <= d;
        end if;
    end process;
end;
```

Visualização do RTL viewer

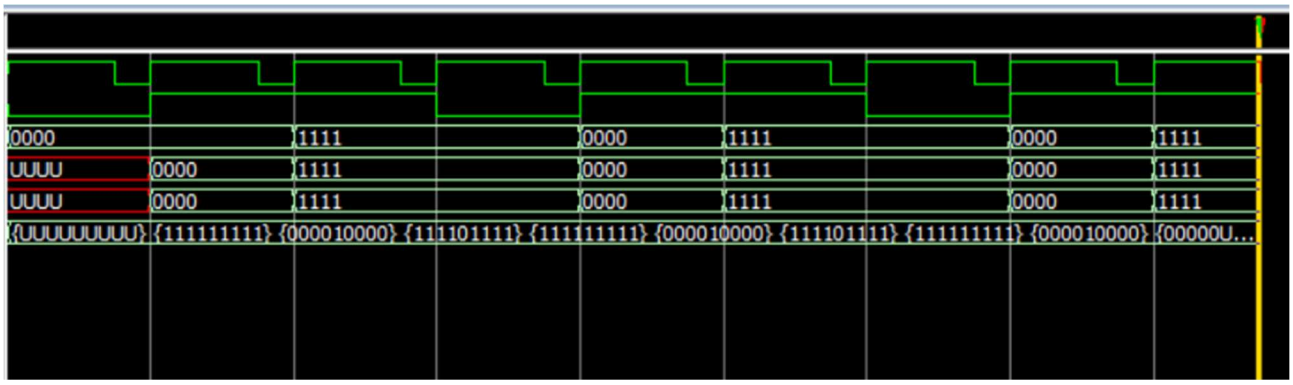


Para simulações com o Modelsim altera construímos um testbench e um tesvector que contém informações de como o circuito deve se comportar

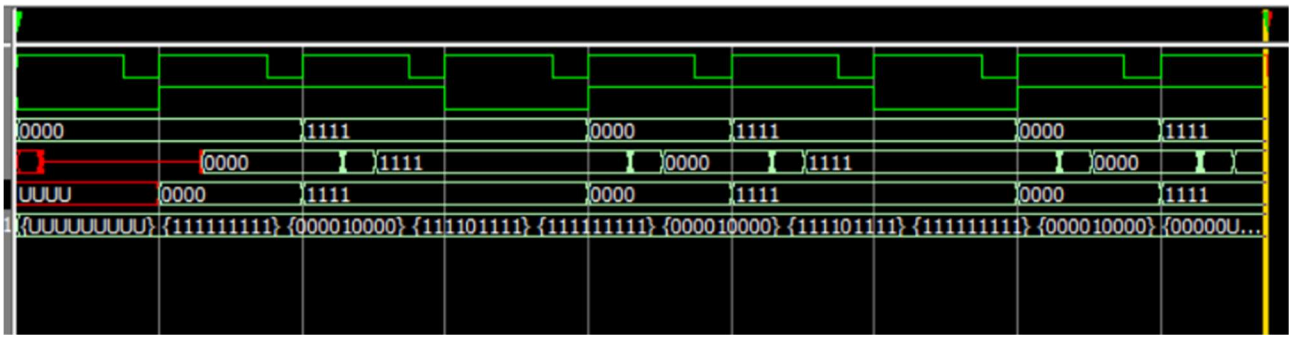
```
1  00000_UUUU
2  00001_0000
3  11111_1111
4  11110_1111
5  00001_0000
6  11111_1111
7  11110_1111
8  00001_0000
9  11111_1111
```

Nota-se, no vetor de teste, que a entrada será copiada pra saída se o clock estiver em ‘1’, caso o clock mude pra ‘0’ a saída será igual a saída anterior.

Simulação RTL level:



Simulação Gate level:



DIRETÓRIOS UTILIZADOS:

floprs (Resettable flip-flop síncrono):

Código VHD: floprs → floprs.vhd
Código testbench: floprs → testbench → testbench_floprs.vhd
Arquivo do testvector: floprs → simulation → modelsim → florsinc.tv

flopenr (Resettable flip-flop com enable):

Código VHD: flopenr → flopenr.vhd
Código testbench: flopenr → testbench → testbench_flopenr.vhd
Arquivo do testvector: flopenr → simulation → modelsim → flopenr.tv

sync (sincronizador):

Código VHD: sync → sync.vhd
Código testbench: sync → testbench → testbench_sync.vhd
Arquivo do testvector: sync → simulation → modelsim → example.tv

latch1 (latch):

Código VHD: latch1 → latch1.vhd
Código testbench: latch1 → testbench → testbench_latch1.vhd
Arquivo do testvector: latch1 → simulation → modelsim → latch.tv